Design of a Rapidly-exploring Random Tree (RRT) Based Path Planner

Shanshan Jiang

***Abstract*—In this report, an RRT based path planner is introduced. Two methods are used to plan the path: RRT and RRT-connect. In RRT-connect method, instead of extending goal tree towards one configuration until trapped, randomly sampling is used to extend both start tree and goal tree for computation simplicity. Once a configuration is randomly sampled, the start tree and the goal tree extend towards this configuration by one step at the same time. In this way, the start tree and the goal tree have more opportunities to meet each other and get out of the obstacle trap. A MATLAB GUI is designed for users to interact with the path planner and set their own configurations.**

***Index Terms*—Robotics, Path, RRT**

I. Introduction

PATH planning is one of the key steps in robotics. In path planning, how to find a collision free path in configuration space is a problem. In complicated configuration space, calculating a path satisfying all the constraints of the obstacles may be computational cost and sometimes cannot get optimal solutions [1]. In such case, finding an available is of more importance than getting the optimal solution [1].

Randomized searching method search with every probability to get a possible solution connecting start and goal configuration given enough time, so it is popular in complicated configuration environment [1]. RRT and RRT-connect are two mainstream randomized searching methods. RRT extends the tree from one point while RRT-connect extends two trees from start and goal point towards each other [1]. The object of this project is to design a system to explain how RRT and RRT-connect work in searching a path and to compare the difference and efficiency between these two methods.

The outline of this project is randomly sampling in configuration space, detecting collision of obstacles, calculating configuration distance of current configuration and the goal, animating the movement from start configuration to goal configuration.

This design working platform is MATLAB 2016b.

II. Guidelines For Running The Program

In this section, a user guide of how to use this system to run RRT path planner and RRT connect path planner is provided. The user guide mainly contains three parts: set working environment (start and goal configuration, obstacles), Path Planner for RRT and RRT connect, other user interactions with the MATLAB GUI.

A. Set Obstacles, Start and Goal

The MATLAB GUI interface is shown in Fig. 1. As the main object of this design is to see how RRT and RRT connect algorithms execute in finding a path. So here the key focus is the animation of RRT based algorithms instead of robot and obstacle shapes since algorithm executing steps are same for more complicated cases. For simplicity, the obstacles and robot are set as circles here.

Set obstacles: user can click directly in the rectangle box to add circle obstacles. The color of obstacles is dark green. User can click button *Set Obstacle Radius* to edit the size of the one obstacle and then click on one obstacle to change its size, as Fig. 2. is shown. User can click button *Remove Obstacles* to remove one or more obstacles.

Set start and goal: user can click *Set Start Point* and *Set End Point* to add robot start and end configuration. The color of the robot is blue. As Fig. 2. is shown.

S. Jiang, is with the Department of Computer Science, University of North Carolina at Charlotte, NC 28223 USA (e-mail: sjiang6@uncc.edu).

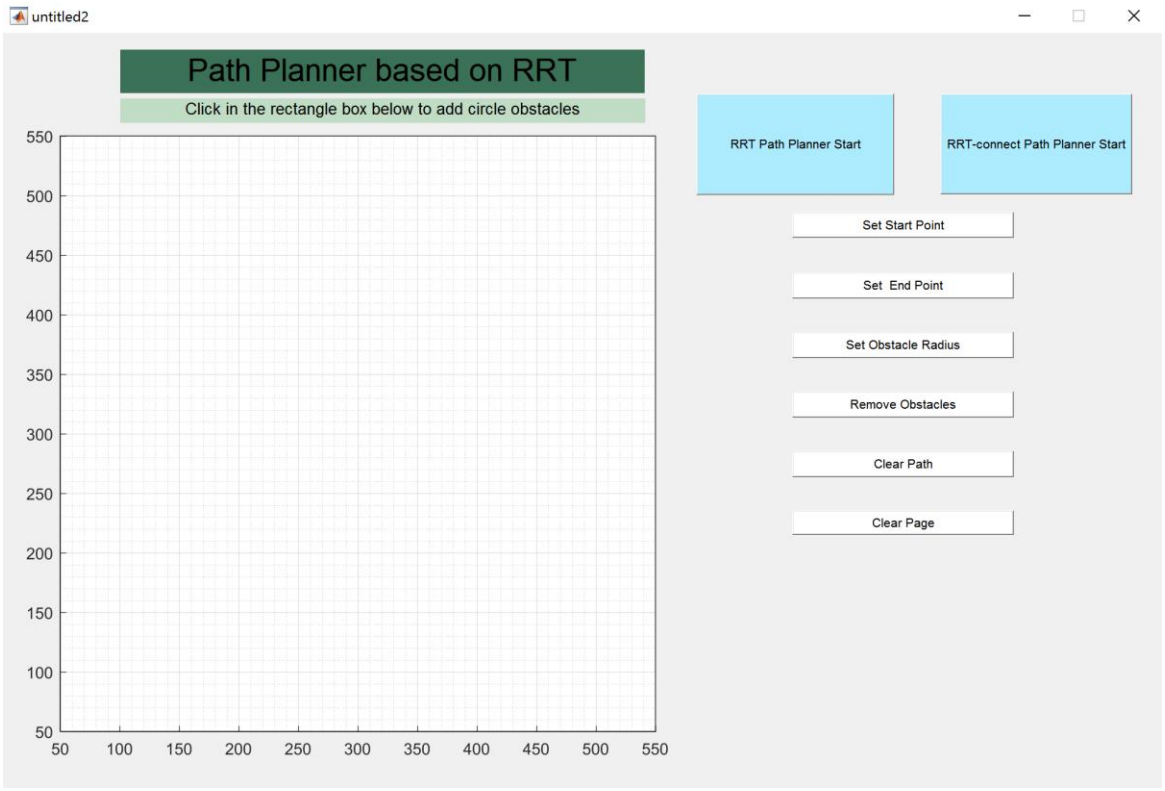

Fig. 1. GUI interface

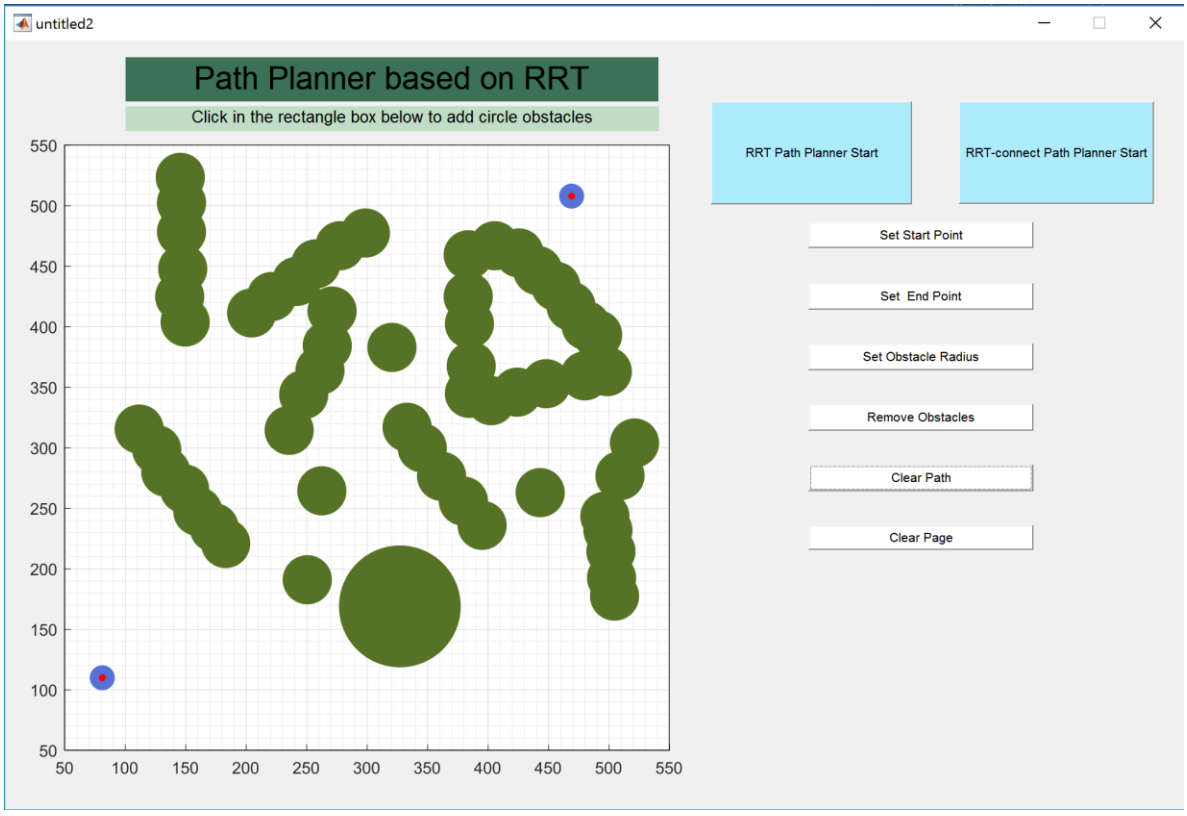

Fig. 2. Add obstacles, start and goal

B. RRT and RRT Connect Path planner

After setting obstacles and robot start and end position. User can click blue button *RRT Path Planner Start* and *RRT-connect Path Planner Start* in Fig. 2.

Case 1: This case is a relative complicate example. The result of RRT is shown in Fig. 3. The result of RRT connect is shown in Fig. 4.

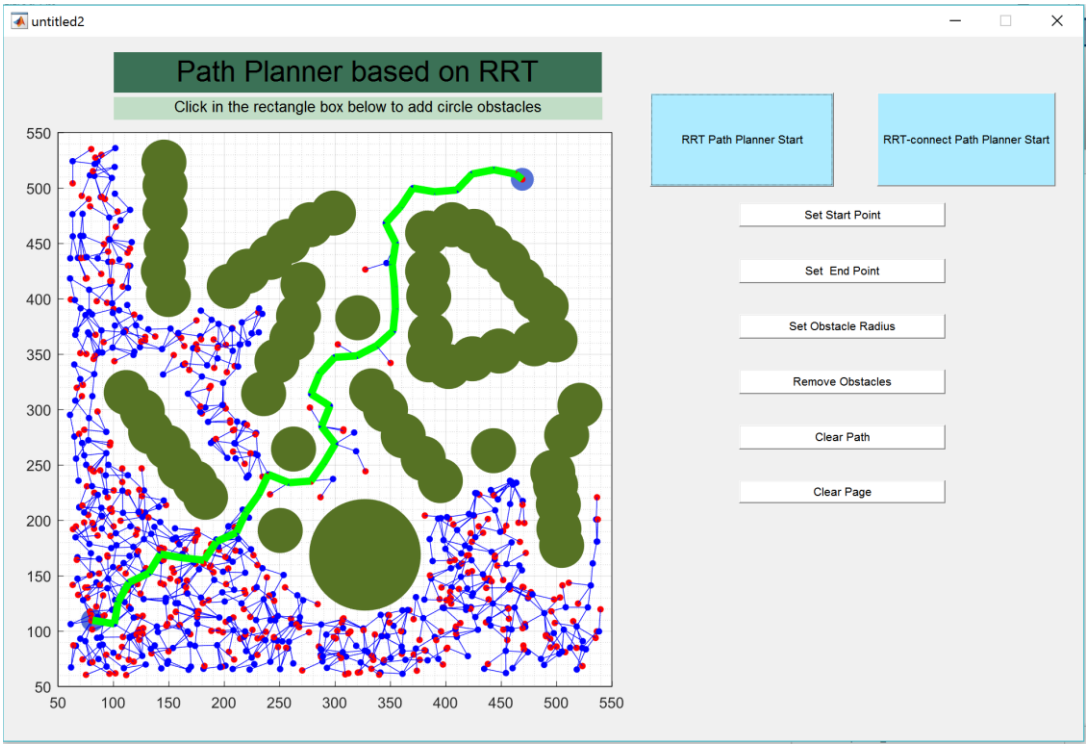

Fig. 3. RRT path planner for case one

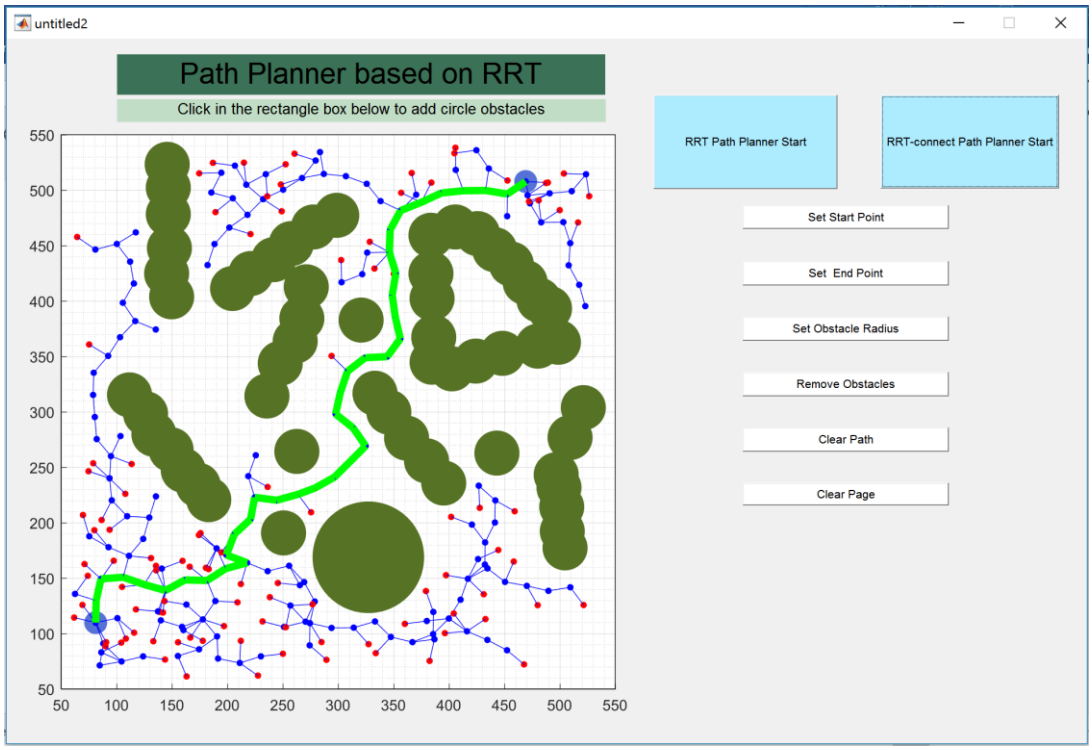

Fig. 4. RRT connect path planner for case one

Case 2: This case is a simple example. The result of RRT is shown in Fig. 5. The result of RRT connect is shown in Fig. 6.

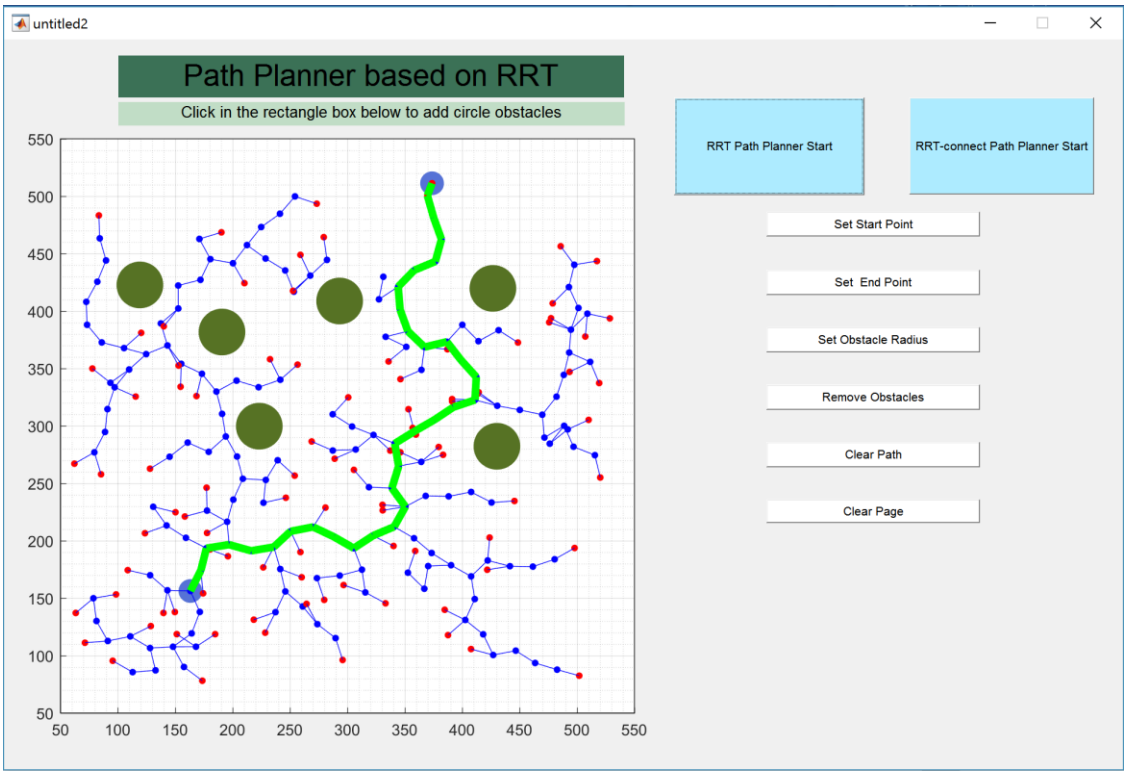

Fig. 5. RRT path planner for case two

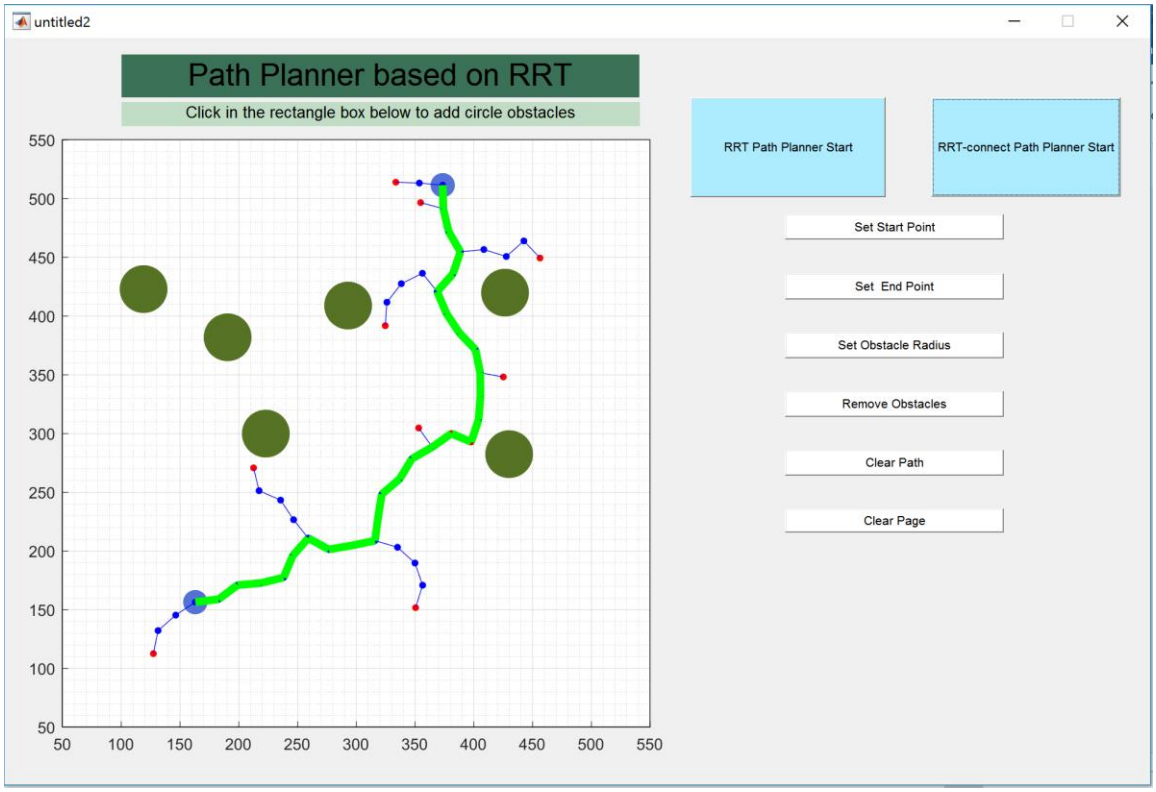

Fig. 6. RRT connect path planner for case two

C. Other Interactions

User can click button *Clear Path* to clear current RRT or RRT connect plotting in order to test another algorithm using same obstacle configuration. User can click button *Clear Page* to clear all obstacles, start and goal point.

III. Analysis of results

A. Performance in Different Working Environment

From above results, RRT and RRT connect will finally find a possible path given enough time if there is a solution. If the working environment is relative complicated, the optimal path is hard to calculate directly. In this case, RRT and RRT connect will show its advantage in path searching, such as Fig. 3 and Fig. 4. If the working environment is very simple, as Fig. 5 and Fig. 5 shows, RRT and RRT connect will be computational cost when optimal path is easy to get using other methods which calculate solution directly.

B. Comparison between RRT and RRT connect

RRT connect method would contain fewer searching steps comparing with RRT in most cases. That is because using RRT, the decision is made by calculating one tree with goal point. While using RRT connect, the decision is made by calculating the distance between two trees extend towards each other. The probability for two trees to meet when grow toward each other is higher than one tree to meet one point.

IV. Possible Future Improvements

As RRT and RRT connect show advantage in complicated working environment, so in the future, we can design more complicated robot and obstacles to highlight this advantage. Also, a smoother path can be optimized to meet practical requirement.

References

[1] Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. Vol. 2. IEEE, 2000.